EEL 6764 001: Graduate Computer Architecture Spring 2025
Instructor: Dr. Srinivas Katkoori
Homework 3
Solutions

1. (10 pts) Data Hazards and Pipeline Timing – Solve problem C.1 on pages C-71 and C-72a)

x1	ld	addi
x1	addi	sd
x2	addi	sub
x2	addi	ld
x2	addi	sd
x4	sub	bnez

b) Forwarding is performed only through the register file. Branch outcomes and targets are not known until the end of the execute stage. All instructions introduced to the pipeline prior to this point are flushed.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ld	x1, 0(x2)	F	D	X	M	W													
daddi	x1, x1, 1		F	s	s	D	X	M	W										
sd	x1, 0(x2)					F	s	s	D	X	M	W							
daddi	x2, x2, 4								F	D	X	M	W						
dsub	x4, x3, x2									F	s	s	D	X	M	W			
bnez	x4, Loop												F	s	s	D	X	M	W
LD R1, 0(R2)																		F	D

Since the initial value of x3 = x2 + 396 and equal instances of the loop adds 4 to x2, the total number of iterations is 99.
Notice that there are eight cycles lost to RAW hazards including the branch instruction. Two cycles are lost after the branch because of the instruction flushing.
It takes 16 cycles between loop instances so, the total number of cycles is $98 \times 16 + 18 = 1586$
The last loop takes two additional cycles since this latency cannot be overlapped with additional loop instances.

c) Now we are allowed normal bypassing and forwarding circuitry. Branch outcomes and targets are known now at the end of decode.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ld	x1, 0(x2)	F	D	X	M	W													
daddi	x1, x1, #1		F	D	s	X	M	W											
sd	x1, 0(x2)			F	s	D	X	M	W										
daddi	x2, x2, #4					F	D	X	M	W									
dsub	x4, x3, x2						F	D	X	M	W								
bnez	x4, Loop							F	s	D	X	M	W						
(incorrect instruction)										F	s	s	s	s					
ld	x1, 0(x2)										F	D	X	M	W				

Again, we have 99 iterations. There are two RAW stalls and a flush after the branch since the branch is taken. The total number of cycles is $9 \times 98 + 12 = 894$
The last loop takes three additional cycles since this latency cannot be overlapped with additional loop instances.

d)

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ld	x1, 0(x2)	F	D	X	M	W													
daddi	x1, x1, #1		F	D	s	X	M	W											
sd	x1, 0(x2)			F	s	D	X	M	W										
daddi	x2, x2, #4					F	D	X	M	W									
dsub	x4, x3, x2						F	D	X	M	W								
bnez	x4, Loop							F	s	D	X	M	W						
ld	x1, 0(x2)									F	D	X	M	W					

Again, we have 99 iterations. We still experience two RAW stalls, but since we correctly predict the branch, we do not need to flush after the branch. Thus, we have only $8 \times 98 + 12 = 796$

e)

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
ld	x1, 0(x2)	F1	F2	D1	D2	X1	X2	M1	M2	W1	W2										
daddi	x1, x1, #		F1	F2	D1	D2	s	s	s	X1	X2	M1	M2	W1	W2						
sd	x1, 0(x2)			F1	F2	D1	D2	s	s	s	X1	X2	M1	M2	W1	W2					
daddi	x2, x2, #4				F1	F2	D1	D2	s	s	s	X1	X2	M1	M2	W1	W2				
dsub	x4, x3, x2					F1	s	s	s	F2	D1	D2	s	X1	X2	M1	M2	W1	W2		
bnez	x4, Loop									F1	F2	D1	s	D2	X1	X2	M1	M2	W1	W2	

ld	x1, 0(x2)	F1	F2	s	D1	D2	X1	X2	M1	M2	W1	W2

We again have 99 iterations. There are three RAW stalls between the ld and addi, and one RAW stall between the daddi and dsub. Because of the branch prediction, 98 of those iterations overlap significantly. The total number of cycles is $10 \times 98 + 19 = 999$

f)

Clock-cycle time = Cycle time for longest stage + Register delay

Hence:

For the 5-stage pipeline,

Clock-cycle time = 0.8 + 0.1 = 0.9 ns

Hence:

For the 10 -stage pipeline,

$Clock - cycle\ time = \frac{0.8}{2} + 0.1 = 0.5ns$

g)

We know that :

$CPI = \frac{Number\ of\ cycles\ in\ Total}{Total\ Instructions\ Executed}$

$Average\ Instruction\ Time = CPI \times Cycle\ Time$

$CPI\ of\ 5 - stage\ Pipeline = \frac{796}{99 \times 6} = 1.34$

$Avg\ Inst\ Exe\ Time\ 5 - stage = 1.34 \times 0.9 = 1.21$

$CPI\ of\ 10 - stage\ Pipeline = \frac{999}{99 \times 6} = 1.68$

$Avg\ Inst\ Exe\ Time\ 10 - stage = 1.68 \times 0.5 = 0.84$

2. 15 pts) Branch Hazards – Solve problem C.2 on page C-72.

a) The performance of ideal pipeline without branch hazards is the pipeline depth, which is 4.

Next, for branch hazards, we need to figure out the stall cycles for each type of branches.

i. For unconditional branches, the stall cycle is 1.
ii. For conditional branches which is taken, the stall cycles is 2.
iii. For conditional branches which is not taken, the stall cycle is 1.

Considering the frequencies of different types of branches,

$The\ performance\ of\ pipeline\ with\ branch\ hazards = \frac{1}{1 + pipeline\ stalls} \times pipeline\ depth$

$= \frac{4}{1 + (1 \times 1\% + 2 \times 15\% \times 60\% + 1 \times 15\% \times 40\%} = 3.2$

$$Speedup = \frac{Pipeline\ Speedup\ withouit\ Hazards}{Pipeline\ Speedup\ with\ Hazards}$$
$$= \frac{4}{3.2} = 1.25$$

b)

The stall cycles for different branches are derived as below.

i. Unconditional Branches : 4
ii. Conditional Branches (taken) : 9
iii. Conditional Branches (not taken) : 8

$$The\ performance\ of\ pipeline\ with\ branch\ hazards = \frac{1}{1 + 4 \times 1\% + 9 \times 15\% \times 60\% + 8 \times 15\% \times 40\%}$$
$$= \frac{15}{2.33}$$

The speedup of the ideal pipeline over the one with branch hazards

$$= \frac{15}{\frac{15}{2.33}} = 2.33$$

3. (15 pts) Deep Pipeline Performance Analysis – Solve C.7 on page C-75

a)

Given 5-stage pipeline experiences a stall due to a hazard every 5 instructions and 12 stage pipeline experiences three stalls every eight instructions.

$$Execution\ Time = IC \times CPI \times Cycle\ Time$$

$$Speedup = \frac{Execution\ Time\ of\ 5-stage\ Pipeline}{Execution\ Time\ of\ 12-stage\ Pipeline}$$

$$= \frac{IC \times \frac{6}{5} \times 1}{IC \times \frac{11}{8} \times 0.6} = 1.45$$

b)

The branch mis predict penalty for the first machine = 2 cycles
The branch mis predict penalty for the second machine = 5 cycles

$$CPI(5-stage) = Original\ CPI\ without\ branch\ mispredict + mispredict\ penalty \times percentage\ of\ branch\ instructions \times mispredict\ rate$$
$$= \frac{6}{5} + 0.20 \times 0.05 \times 2$$
$$= 1.22$$
$$CPI\ (12-stage) = \frac{11}{8} + 0.20 \times 0.05 \times 5 = 1.425$$

$$Speedup = \frac{IC \times 1.22 \times 1}{IC \times 1.425 \times 0.6} = 1.42$$

4. **(20 pts) The following series of branch outcomes occurs for a single branch in a program. (T means the branch is taken, N means the branch is not taken).**
   Index 1 2 3 4 5 6 7 8 9 10 11 12 13
   T, T, N, T, N, T, T, T, T, N, T, T, N
   a) Assume that we are trying to predict this sequence with a Branch History Table (BHT) using a 1-bit prediction. The counters of the BHT are initialized to the N state. Which of the branches would be mis predicted? Show their indices.
   1-bit predictor

   Red slots on third row show 8 total mis-prediction

Index	1	2	3	4	5	6	7	8	9	10	11	12	13
	T	T	N	T	N	T	T	T	T	N	T	T	N
predictor	N	T	T	N	T	N	T	T	T	T	N	T	T

   a) Repeat the above exercise with a 2-bit predictor as shown in Figure C.15 initialized to 10.

   2-bit predictor

   Red slots on third row show 4 total mis-prediction

Index	1	2	3	4	5	6	7	8	9	10	11	12	13
	T	T	N	T	N	T	T	T	T	N	T	T	N
predictor	10	11	11	10	11	10	11	11	11	11	10	11	11

5. **10 pts) Performance Evaluation – Solve 3.1 on page 266.**

The baseline performance (in cycles, per loop iteration) of the code sequence in Figure 3.47, if no new instruction's execution could be initiated until the previous instruction's execution had completed is 37. Each instruction requires one clock cycle of execution (a clock cycle in which that instruction, and only that instruction, is occupying the execution units; since every instruction must execute, the loop will take at least that many clock cycles). To that base number, we add the extra latency cycles. Also remember the additional cycle for branch delay slot.

6. **(10 pts) Ideal Dependency Detection – Solve 3.2 on page 267**
   How many cycles would the loop body in the code sequence in Figure 3.47 require if the pipeline detected true data dependencies and only stalled on those, rather than blindly stalling everything just because one functional unit is busy? → We need 26, as shown below. Remember, the point of the extra latency cycles is to allow an instruction to

complete whatever actions it needs, in order to produce its correct output. Until that output is ready, no dependent instructions can be executed. So, the first fld must stall the next instruction for three clock cycles. The fmul.d produces a result for its successor, and therefore must stall 4 more clocks, and so on.

```
Loop:   fld      f2,  0(Rx)
        <stall   fld>
        <stall   fld>
        <stall   fld>
        fmul.d f2, f0, f2
        <stall   fmul>
        <stall   fmul>
        <stall   fmul>
        <stall   fmul>
        fdiv.d   f8, f2, f0
        fld   f4, 0(Ry)
        <stall   fld>
        <stall   fld>
        <stall   fld>
        fadd.d f4, f0, f4
        <stall   fdiv>
        <stall   fdiv>
        <stall   fdiv>
        <stall   fdiv>
        <stall   fdiv>
        fadd.d f10, f8, f2
        fsd  f4, 0(Ry)
        addi     Rx, Rx, 8
        addi     Ry, Ry, 8
        sub x20, x4, Rx
        bnz x20, Loop
```

7) 20 pts) Dynamic Scheduling Draw the basic structure of a RISC-V floating point unit for Tomasulo's algorithm. Explain how code is executed with an example

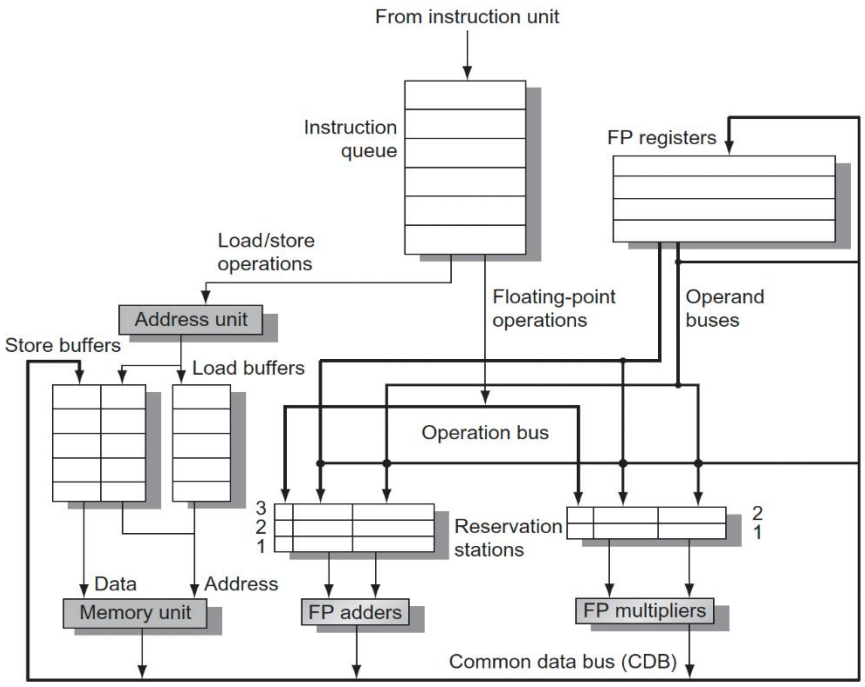

Fig 1: The basic structure of a RISC-V floating -point unit using Tomasulo's Algorithm
Example of a code sequence – (You can consider example of your choice)

1. fld f6, 32(x2)
2. fld f2, 44 (x3)
3. fmul. d f0, f2, f4
4. fsub. d f8, f2, f6
5. fdiv. d f0, f0, f6
6. fadd. d f6, f8, f2

One of the advantages of using Tomasulo's Algorithm is the elimination of WAW and WAR hazards, is accomplished by renaming registers using the reservation stations and by process of storing operands into the reservation station as soon as they are available. The code sequence issues both the fdiv. d and the fadd. d even though there is a WAR hazard involving f6. The hazard is eliminated in one of two ways. First, if the instruction provides the value for the fdiv. d has completed, then Vk will store the result, allowing fdiv. d to execute independent of the fadd. d.

On the other hand, if the fld hasn't completed, then Qk will point to the Load1 reservation station, and the fdiv.d instruction will be independent of the fadd. d.

Any uses of the result of the fdiv. d will point to the reservation station, allowing the fadd. d to complete and store its value into the registers without affecting the fdiv. d

		Instruction status		
Instruction		Issue	Execute	Write result
fld	f6,32(x2)	√	√	√
fld	f2,44(x3)	√	√	
fmul.d	f0,f2,f4	√		
fsub.d	f8,f2,f6	√		
fdiv.d	f0,f0,f6	√		
fadd.d	f6,f8,f2	√		

	Reservation stations						
Name	Busy	Op	Vj	Vk	Qj	Qk	A
Load1	No						
Load2	Yes	Load					44 + Regs[x3]
Add1	Yes	SUB		Mem[32 + Regs[x2]]	Load2		
Add2	Yes	ADD			Add1	Load2	
Add3	No						
Mult1	Yes	MUL		Regs[f4]	Load2		
Mult2	Yes	DIV		Mem[32 + Regs[x2]]	Mult1		

	Register status								
Field	f0	f2	f4	f6	f8	f10	f12	…	f30
Qi	Mult1	Load2		Add2	Add1	Mult2			

Fig:2 Reservation stations and register tags shown when all of the instructions have been issued but only the first load instruction has completed and written its results to the CDB.

The second load has completed effective address calculation but is waiting on the memory unit. The fadd.d instruction , which has a WAR hazard at the WB stage, has issued and could complete before the fdiv.d initiates.